REMARKS

The claims remaining in the present application are Claims 1-16. Claim 16 has been added. Claims 7 and 12 have been amended. The Title of the Specification and the Instant Specification have been amended. No new matter has been added as a result.

SPECIFICATION

The Title of the Specification has been amended. The Instant Specification has been amended at the paragraph beginning at page 9, line 9 to add an application serial number to a referenced US Patent Application.

35 U.S.C. §112

Claim 7 is rejected under 35 U.S.C. §112,¶2 for an informality. Claim 7 has been amended to correct the informality. It is respectfully submitted that amended Claim 7 complies with 35 U.S.C. §112,¶2. Claim 12 has been amended to correct an informality. Review and approval of these amendments are respectfully requested.

35 U.S.C. §102

Claims 1-15 are rejected under 35 U.S.C. §102(e) as being anticipated by Lethin et al., U.S. Patent Application Publication 2002/0,147,969 (hereinafter, Lethin). The rejection is respectfully traversed under the following rationale.

Claim 1

Claim 1 recites:

> A method of transferring between types of conversion processes in a computer which converts instructions from a target instruction set to a host instruction set comprising the steps of:
> executing code morphing software including an interpreter and a translator to generate host instructions from target instructions,
> detecting at intervals whether the interpreter or the translator is operating,
> increasing a count if the interpreter is operating and decreasing the count if the translator is operating, and
> changing from interpreting to translating a sequence of target instructions when the count reaches a selected maximum.

Claim 1 recites the limitation of "increasing a count if the interpreter is operating and decreasing the count if the translator is operating." Thus, Claim 1 recites updating a counter based on whether the interpreter is operating or the translator is operating.

As discussed below, Lethin fails to disclose this limitation. In contrast, Lethin discloses a mechanism which is used to monitor compiler utilization. Specifically, the mechanism in Lethin compares requests for compilation input to the compiler to compiled instructions output by the compiler. Applicants note that Lethin does not increase a count if the interpreter is operating, as claimed. In fact, as the mechanism in Lethin depends on requests to the compiler and outputs from the compiler, it does not measure interpreter utilization. That is, the request for compilation does not indicate interpreter usage, but rather a decision not to use the interpreter. Lethin recites, "the interpreter task counts how often a branch instruction jumps to a particular destination address. When the count passes a threshold, the interpreter sends a translation request including the destination address" (paragraph [0626]).

Applicants note that the rejection is considering the translation request to be the count claimed in Claim 1. Applicants further note that the translation request only occurs if the threshold is passed. Thus, the translation request is not a measure of interpreter usage, as claimed. Rather, it is a measure of a decision to use the compiler.

Claim 1 further recites the limitation of, "changing from interpreting to translating a sequence of target instructions when the count reaches a selected maximum." Applicants note that the count is based on interpreter utilization versus compiler utilization, as claimed. However, the count is not based on counting the number of times a sequence of instructions is executed.

Lethin fails to disclose "changing from interpreting to translating a sequence of target instructions when the count reaches a selected maximum." In contrast, Lethin discloses that the decision to change from interpreting to compiling a sequence of instructions is made based on how many times the sequence of instructions has been executed. That is, when the threshold is passed, indicating a sequence of instruction has been executed a given number of times, the mode is changed, as indicated in the previously cited passage.

In contrast to disclosing the claimed limitation of "changing from interpreting to translating a sequence of target instructions when the count reaches a selected maximum," Lethin discloses that the threshold is changed based on the result of the comparison of the requests to compile versus the output compiled instructions.

Changing the threshold does not change from interpreting to translating a sequence of target instructions. Rather, Lethin requires that a separate count of how many times a particular sequence of instructions have been executed is maintained and only changes from interpreting to compiling if that count of the number of times a particular sequence of instructions has been executed exceeds the threshold.

For the foregoing rationale, it is respectfully submitted that Claim 1 is not anticipated Lethin. As such, allowance of Claim 1 is respectfully submitted.

Claims 2-9 depend from Claim 1, which are respectfully believed to be allowable. As such, allowance of Claims 2-9 is earnestly solicited.

CLAIM 10

Claim 10 recites:

> A method of optimizing execution by a computer which dynamically converts instructions from a target instruction set to a host instruction set comprising the steps of:
> providing a plurality of instruction conversion processes each providing a different level of optimization for converted instructions from a target instruction set to a host instruction set,
> providing means for determining dynamically which conversion process best converts each sequence of instructions, and
> converting a sequence of instructions using a conversion process determined to best convert the sequence of instructions.

Claim 10 recites the limitation of providing means for determining dynamically which conversion process best converts each sequence of instructions. It is respectfully asserted that Lethin fails to disclose this claimed limitation.

The rejection cites element 144, the block picker, as a means of determining dynamically which conversion process best converts each sequence of instructions. Applicants respectfully submit that the block picker 114 is used to choose a segment of the code to compile, creates a control flow graph (CFG) that describes the original instructions to compile, and passes the CFG to the block layout unit. However, the block picker 114 does not determine which conversion process should be used. As such, the block picker is not understood by Applicant to teach or suggest a means for determining dynamically which conversion process best converts each sequence of instructions, as claimed.

For the foregoing rationale, it is respectfully submitted that Claim 10 is not anticipated Lethin. As such, allowance of Claim 10 is respectfully submitted.

Claims 11-15 depend from Claim 10, which are respectfully believed to be allowable. As such, allowance of Claims 11-15 is earnestly solicited.

NEW CLAIM

Claim 16 has been added. It is respectfully asserted that New Claim 16 is neither taught nor suggested by the cited references. As such, allowance of New Claim 16 is respectfully solicited.

CONCLUSION

In light of the above listed amendments and remarks, reconsideration of the rejected Claims is requested. Based on the arguments and amendments presented

above, it is respectfully submitted that Claims 1-16 overcome the rejections of record and, therefore, allowance of Claims 1-16 is earnestly solicited. Should the Examiner have a question regarding the instant amendment and response, the Applicants invite the Examiner to contact the Applicants' undersigned representative at the below listed telephone number.

Respectfully submitted,
WAGNER, MURABITO & HAO LLP

Dated: 8/12, 2003

Ronald M. Pomerenke
Registration No. 43,009

Address: WAGNER, MURABITO & HAO LLP
Two North Market Street
Third Floor
San Jose, California 95113

Telephone: (408) 938-9060 Voice
(408) 938-9069 FAX